Enhancing Privacy via Hierarchical Federated Learning

Aidmar Wainakh
Telecooperation Lab
Technical University of Darmstadt
wainakh@tk.tu-darmstadt.de

Alejandro Sanchez Guinea
Telecooperation Lab
Technical University of Darmstadt
sanchez@tk.tu-darmstadt.de

Tim Grube
Telecooperation Lab
Technical University of Darmstadt
grube@tk.tu-darmstadt.de

Max Mühlhäuser
Telecooperation Lab
Technical University of Darmstadt
max@tk.tu-darmstadt.de

arXiv:2004.11361v1 [cs.CR] 23 Apr 2020

Abstract—**Federated learning suffers from several privacy-related issues that expose the participants to various threats. A number of these issues are aggravated by the centralized architecture of federated learning. In this paper, we discuss applying federated learning on a hierarchical architecture as a potential solution. We introduce the opportunities for more flexible decentralized control over the training process and its impact on the participants' privacy. Furthermore, we investigate possibilities to enhance the efficiency and effectiveness of defense and verification methods.**

Index Terms—**Federated learning, hierarchical architecture, privacy enhancement**

1. Introduction

In recent years, data breaches have dramatically increased [16]. At the same time, more services are making use of large amounts of personal data as part of machine learning implementations to provide value to users. These two factors have contributed to making users of services increasingly concerned about their privacy. The concept of Federated Learning (FL) has been proposed to alleviate this issue by allowing multiple users to build a joint model without sharing their data, under the coordination of a central server [17]. However, FL's distributed training process, together with its strong dependence on a central server, have increased the attack surface against users yet again [11]. Fully decentralized learning based on peer-to-peer topology has been proposed to cope with the issues associated with having a central server, such as a performance bottleneck and a single point of failure [27]. However, the convergence time and robustness against user churn remain open challenges.

A hierarchical architecture lies in between the two extremes, centralized and fully decentralized architectures, as a solution able to cope with complexity, scalability, and failure-related issues [9]. Applying FL based on a hierarchical architecture is referred to as Hierarchical Federated Learning (HFL). HFL naturally matches emerging decentralized infrastructures (e.g., edge and fog computing) and heterogeneous nature of real-world systems [13]. Most importantly, HFL provides a unique capacity for improving privacy. As such, it reduces the centralization of power

and control in the hands of the central server. In addition, HFL allows flexible placement of defense and verification methods within the hierarchy and enables these methods to be applied more efficiently and effectively. Furthermore, HFL creates the possibility to employ the trust between users to mitigate a number of threats.

In this paper, we explore the potential benefits of using HFL to address privacy-related issues currently discussed in the FL-community. The rest of this paper is organized as follows. In Section 2, we present FL and three of its privacy-related issues. Then, we introduce, in Section 3, the concept of HFL and potential applications. In Section 4, we discuss the privacy implications of HFL. Finally, we wrap up with a conclusion section.

2. Federated Learning and Open Issues

Federated Learning (FL) is a machine learning setting where multiple entities (users) train a joint model. Each user trains the model locally on their data and shares only the model updates with a central server. The central server aggregates the updates from the users into a new global model. This setting mitigates a number of privacy risks that are typically associated with conventional machine learning, where all training data should be collected, then used to train a model [11]. In spite of the various benefits of FL, several privacy-related issues are still open. A number of these issues are mainly derived from the centralized nature of the FL architecture. Next, we describe three of these issues, namely centralized control, limited verifiability, and constrained defenses.

Centralized control. The server in FL plays the role of a central coordinator that performs the following core functions: (1) sampling users, i.e., selecting which users participate in the training process, (2) broadcasting the model and training algorithm, (3) aggregating the model updates, and (4) broadcasting the updated global model. Although such a setting limits the computational cost for the users, it places all the control on a single party, the server. Thus, the server potentially represents a performance bottleneck and a single point of failure. Furthermore, from a privacy perspective, concentrating such functions on the server leaves the users with limited or no control over the process. Consequently, the users have

to rely on trusting the server to perform all the functions correctly while maintaining best privacy practices to protect their model updates. In this respect, if the server were to be malicious or an adversary were to compromise the server, various attacks could be perpetrated against the users, e.g, reconstruction attacks [30], [33]

Limited verifiability. In FL, the server and users perform several computations locally and share the results with each other. The users share their updates, while the server shares the aggregated model. In order to allow the server and users to prove to each other that they perform the expected computations correctly and share a legitimate output, the computations need to be verified. Multiple approaches based on zero-knowledge proofs have been proposed to apply this verification [11]. However, these approaches suffer from two main limitations. First, the types of the proofs provided are limited (e.g., range proofs [3]). Second, the time required for verification grows typically exponentially with the number of users [4], [31], which renders these approaches unscalable, and thus unsuitable for large-scale applications. Another technique to tackle the verification issue is to use a Trusted Execution Environment to perform the computations [11]. However, these environments are not widely available on user devices, especially smartphones.

Constrained defenses. Although FL provides improvements for users' privacy, it opens the door for plenty of attacks, which can be applied by both malicious servers and malicious users. In this work, we focus on privacy attacks, where the attacker aims at inferring information about users. These attacks occur in two modes: passive and active.

- *Passive attacks*: The attacker observes the joint model and periodic updates, which can be used to infer information about other users. Shokri et al. [23] introduce a membership inference attack using shadow models, which can be performed in FL. Melis et al. [19] propose a property inference attack, leveraging snapshots of the global model. Zhou et al. [33] obtain the training data of a target user from their shared updates (gradients). To mitigate such attacks, several defense techniques can be applied by the server and users. The users can perturb their model updates before sharing them by using one or a combination of the following methods: (1) adding noise (e.g., local differential privacy [18]), (2) sharing only a fraction of the updates [22], or (3) using the dropout technique in a neural network [25]. However, these techniques usually lead to a substantial loss of the model accuracy. That is, the trade-off between privacy and accuracy needs to be considered [12]. On the server side, protecting user privacy requires to break the linkability of the individual users with their updates. To achieve this, a secure computation can be performed, e.g, secure aggregation [2] or secure shuffling [1]. Besides the efficiency issues these techniques suffer from, they hinder the server from detecting malicious updates, creating by that an attack surface for malicious users.
- *Active attacks*: The attacker participates in the training process as a user, who maliciously modifies their updates to infer information about other users. Hitaj et al. [8] propose a reconstruction attack where the attacker provokes the target user to overfit the model on their training data. Melis et al. [19] present a property inference

attack based on multi-task learning. Nasr et al. [20] use gradient ascent to perform a membership inference attack. To detect and mitigate such attacks, several methods can be considered, such as anomaly detectors [21], median-based aggregation [32], trimmed mean [32], and redundancy-based encoding [6]. However, some of these methods require accessing the individual user updates [21], which is impeded by secure aggregation. In addition, the heavy computations and uncertainty of convergence remain main limitations [5].

3. Hierarchical Federated Learning

FL is, by definition, coordinated by a central server [11]. We relax this definition to apply FL within a hierarchical architecture and refer to it as *Hierarchical Federated Learning (HFL)*. A restricted version of HFL, considering only three layers, has been presented in [13]–[15]. However, privacy was not considered a prime goal. In HFL, there is one *root server*, connected to multiple *group servers*, which are organized in a tree structure (see Figure 1). The lowest layer of group servers connects to users (data owners), which are clustered in *user groups*. The hierarchy can contain multiple layers of group servers and can be unbalanced such that different branches vary in their number of layers. The users send their model updates to the upper layer to be aggregated by the corresponding group servers. The aggregation process continues in multiple stages (on each layer) towards the root server. Every two subsequent layers can have a different number of communication (aggregating) rounds before pushing their models to the upper layer. After aggregation, the global model is forwarded along the hierarchy downwards to the users.

From an abstract perspective, a hierarchy is a hybrid solution between centralized and fully decentralized architectures. On the one hand, a hierarchy can cope with the scalability and failure tolerance limitations of centralized architectures. That is evidenced by the existence and active operation of hierarchical architectures within complex, large-scale systems such as the Internet [26]. On the other hand, a hierarchy helps to tackle the management challenges encountered in fully decentralized architectures. We can observe this advantage in the organizational structure of companies [9].

FL was originally proposed with the intention to be based on a cloud computing infrastructure [17]. Such centralized infrastructure represents a network bottleneck, especially with the increase of resource-constrained devices (e.g., smartphones and IoT sensors). To alleviate this issue, there is a trend towards decentralized computing infras-
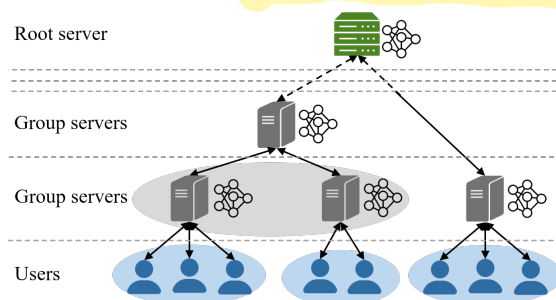

Figure 1. Example of Hierarchical Federated Learning architecture.

tructures (e.g., edge and fog computing) [7], which are introducing hierarchies in communication and computing architectures. HFL represents a natural step in extending FL into such emerging infrastructures.

Online social networks represent yet another application where the system architecture matches with HFL. Motivated by privacy-related issues, more decentralized social networks are emerging [29]. Diaspora and Mastodon are two of the most popular, which leverage the concept of local servers. Users can choose which local server to connect to and where to store their data. This architecture represents a hierarchy where some functions are delegated to the higher-layer servers. In contrast, privacy-related functions (e.g., data storage) are pushed down in the hierarchy to local servers and users.

4. Privacy Implications of HFL

In this section, we discuss the implications of HFL in relation to the privacy-related issues of Section 2.

Centralized control. In FL, the distribution of functions between the server and users is unbalanced; the control is concentrated on the server. In contrast, HFL allows different FL functions to be distributed throughout the hierarchy to the group servers and users, which helps to overcome the performance bottleneck and single point of failure typically found in FL. This is because the group servers can use the local aggregated models within the hierarchy until they converge on the root server at the top. Next, we introduce alternative placements for each of the FL functions.

- *Sampling users*: In FL, the server selects the users to participate in the training process [17]. A malicious server could misuse this privilege to attack specific users [30]. In contrast, HFL allows sampling to be performed in multiple stages, where a lower layer is sampled by the upper one, till the top. This can cope with the gap of trust between the users and server in FL, since each two subsequent layers in HFL are more likely to have a better base for trust and accountability. In fog computing, users can be sampled by edge devices, which can be, for instance, a router in the neighborhood. In decentralized social networks, the local server, which is chosen by the users, can perform sampling. Also, users can apply privacy-preserving distributed sampling algorithms, such as Anonymous Random Walk used in [28].
- *Broadcasting the training algorithm and model*: Congzheng et al. [24] have shown that malicious training algorithms can leak information about the training data. To address this issue in HFL, the group servers can be seen as a protective layer where the algorithms are inspected and verified before being pushed to the users. From an efficiency perspective, HFL provides the possibility for the group servers to adapt the training process to suit the resources of their users' devices, e.g., by adjusting the hyper-parameters of the algorithm or by broadcasting a fraction of the model (sub-model). In FL, training a sub-model by the users can reveal information about their data to the server [11], while in HFL dividing the model to sub-models can happen in different phases through the layers such that an upper layer is not aware how or whether the lower layer divides the model further.

- *Aggregating the model updates*: As opposed to FL, where the aggregation is conducted only by the server, in HFL, multiple levels of aggregation occur according to the number of layers in the hierarchy. This aggregation scheme implies that only the lowest layer of group servers receive individual updates from users, while all the upper layers process aggregated updates. Consequently, the need for secure aggregation in the upper layers is reduced, which can be interpreted to a performance improvement. In addition, the cascade aggregation process helps to protect the identities of the users from the higher-layer servers.
- *Broadcasting the updated global model*: Some privacy attacks in FL require distributing a poisoned model to the target users [8]. The flat topology at the users level in FL allows this type of attacks easy access. While HFL provides the possibility to control the model dissemination more effectively through the multiple layers. In case a poisoned model were to be detected within the hierarchy, the group servers could refrain from pushing it to the users.

Limited verifiability. The poor scalability of the verification methods in FL (e.g., [4], [31]) hinders their deployment in large-scale applications. In HFL, verifying the updates of the users can be performed by the corresponding group servers. Grouping users in comparably small groups reduces the computational overhead on the group servers and makes the deployment of verification methods more feasible.

Constrained defenses. Most of the defense methods in FL come with the cost of lower model accuracy [22], considerable computational overhead [32], or both [6]. HFL provides the possibility to apply these methods in a more flexible manner. For instance, applying different methods in different parts of the hierarchy, such that not all the users suffer from a heavy loss in accuracy nor have to perform costly computations.

As the upper layers in HFL process only aggregated models, not individual users' updates, the necessity of secure aggregation can be reduced or even eliminated in these layers. This can lead to a performance improvement and allows applying anomaly detection methods (e.g., [21]) to detect malicious models received from lower layers. Detecting a malicious model from a specific group server can be followed by excluding that server from further aggregation rounds while maintaining the training process functioning normally in the rest of the hierarchy. Furthermore, the affected server can be notified about its malicious model so it can take more computational expensive countermeasures (e.g., [32]) in their local user group.

Unlike FL, HFL allows leveraging the trust between users as an additional line of defense. In social networks, grouping users based on trusted graphs [10] can reduce the probability of attacks within groups, as it is assumed to be more difficult for malicious users to break into the groups. Consequently, users may relax their local defenses (e.g., reduce the noise added locally) to achieve a model with higher accuracy. At the same time, the group servers can take the responsibility of protecting their groups from attacks across groups by applying specific defense methods (e.g., adding noise to the aggregated updates).

5. Conclusion

In this work, we presented a number of privacy-related issues in Federated Learning (FL), such as centralized control and constrained defenses. We discussed how Hierarchical Federated Learning (HFL) can potentially address these issues. HFL utilizes an architecture that lies between the centralized (FL) and fully decentralized learning and comes with a series of privacy advantages. HFL enables high flexibility in functionality distribution, which facilitates known defense and verification methods. HFL allows leveraging the trust between the participants in different application scenarios to mitigate several threats. Based on this contribution, we wish to encourage further research into the direction of HFL to unlock its full potential.

Acknowledgment

References

[1] Andrea Bittau, Ulfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 441–459, 2017.

[2] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[3] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 315–334. IEEE, 2018.

[4] Lukas Burhhalter. Robust Secure Aggregation for Privacy-Preserving Federated Learning with Adversaries. In *2019 cheeeeck*. ACM, 2019.

[5] Lingjiao Chen, Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. Draco: Byzantine-resilient distributed training via redundant gradients. *arXiv preprint arXiv:1803.09877*, 2018.

[6] Deepesh Data, Linqi Song, and Suhas Diggavi. Data encoding for byzantine-resilient distributed optimization. *arXiv preprint arXiv:1907.02664*, 2019.

[7] Julien Gedeon, Jens Heuschkel, Lin Wang, and Max Mühlhäuser. Fog computing: Current research and future challenges. *KuVS-Fachgespräch Fog Comput*, 1:1–4, 2018.

[8] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 603–618, 2017.

[9] Elliott Jaques. In praise of hierarchy. *Markets, hierarchies and networks: The coordination of social life*, pages 48–52, 1991.

[10] Wenjun Jiang, Guojun Wang, and Jie Wu. Generating trusted graphs for trust evaluation in online social networks. *Future generation computer systems*, 31:48–58, 2014.

[11] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[12] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.

[13] Tao Lin, Sebastian U Stich, Kumar Kshitij Patel, and Martin Jaggi. Don't use large mini-batches, use local sgd. *arXiv preprint arXiv:1808.07217*, 2018.

[14] L Liu, J Zhang, S Song, and KB Letaief. Client-edge-cloud hierarchical federated learning. In *Proc. IEEE Int. Conf. Commun.(ICC), to be published*, 2019.

[15] Lumin Liu, Jun Zhang, SH Song, and Khaled B Letaief. Edge-assisted hierarchical federated learning with non-iid data. *arXiv preprint arXiv:1905.06641*, 2019.

[16] David McCandless. World's Biggest Data Breaches & Hacks. https://informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks, April 2019. Online; accessed 05.03.2020.

[17] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

[18] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private language models without losing accuracy. *arXiv preprint arXiv:1710.06963*, 2017.

[19] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 691–706. IEEE, 2019.

[20] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks. *arXiv preprint arXiv:1812.00910*, 2018.

[21] Shiqi Shen, Shruti Tople, and Prateek Saxena. Auror: Defending against poisoning attacks in collaborative deep learning systems. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 508–519, 2016.

[22] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321, 2015.

[23] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[24] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. Machine learning models that remember too much. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 587–601, 2017.

[25] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[26] Lakshminarayanan Subramanian, Sharad Agarwal, Jennifer Rexford, and Randy H Katz. Characterizing the internet hierarchy from multiple vantage points. In *Proceedings. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 618–627. IEEE, 2002.

[27] Hanlin Tang, Xiangru Lian, Ming Yan, Ce Zhang, and Ji Liu. D ²: Decentralized training over decentralized data. *arXiv preprint arXiv:1803.07068*, 2018.

[28] Aidmar Wainakh, Tim Grube, Jörg Daubert, and Max Mühlhäuser. Efficient privacy-preserving recommendations based on social graphs. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 78–86, 2019.

[29] Aidmar Wainakh, Tim Grube, and M Max. Tweet beyond the Cage: A Hybrid Solution for the Privacy Dilemma in Online Social Networks. In *IEEE Global Communications Conference*, 2019.

[30] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 2512–2520. IEEE, 2019.

[31] Guowen Xu, Hongwei Li, Sen Liu, Kan Yang, and Xiaodong Lin. Verifynet: Secure and verifiable federated learning. *IEEE Transactions on Information Forensics and Security*, 15:911–926, 2019.

[32] Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. *arXiv preprint arXiv:1803.01498*, 2018.

[33] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*, pages 14747–14756, 2019.